A Cartesian Cut Cell Method for Shallow Water Flows with Moving Boundaries

A Cartesian cut cell method for shallow water flows with moving boundaries.

D.M. Causon [a,1] D.M. Ingram [a] C.G. Mingham [a]

[a] *Centre for Mathematical Modelling and Flow Analysis, Manchester Metropolitan University, Manchester, M1 5GD, United Kingdom.*

Abstract

A new computational method for the calculation of shallow water flows with moving physical boundaries is presented. The procedure can cope with shallow water problems having arbitrarily complex geometries and moving boundary elements. Although the method provides a fully boundary-fitted capability, no mesh generation is required in the conventional sense. Solid regions are simply cut out of a background Cartesian mesh with their boundaries represented by different types of cut cell. Moving boundaries are accommodated by up-dating the local cut cell information on a stationary background mesh as the boundaries move. No large-scale re-meshing is required. For the flow calculations, a multi-dimensional high resolution upwind finite volume scheme is used in conjunction with an efficient approximate Riemann solver at flow interfaces, and an exact Riemann solution for a moving piston at moving boundary elements. The method is validated for test problems that include a ship's hull moving at supercritical velocity and two hypothetical landslide events where material plunges laterally into a quiescent shallow lake and a fiord.

Key words: Cartesian cut cell. Shallow water equations. Riemann based schemes. Finite volume method. Moving boundaries.

1 Introduction

It is well-known that the shallow water equations provide a satisfactory model of a variety of physical phenomena such as tidal flows, tsunami propagation, dam-breaks, hydraulic jumps in open channels or the propagation of sharp fronts in the atmosphere. Hence, solutions of the shallow water equations are

[1] Corresponding author. Tel: +44-161-247-3557; fax: +44-161-247-1483; e-mail D.M.Causon@mmu.ac.uk

of considerable practical importance. Like the Euler equations in gas dynamics, the inviscid shallow water equations are a set of nonlinear hyperbolic equations which admit discontinuities in addition to smooth or classical solutions. Therefore, the use of modern upwind schemes is desirable to resolve both discontinuous and smooth solutions without generating spurious oscillations around discontinuities. Examples of such schemes can be found in [1] which employed a Roe-type approximate Riemann solution of the 2-D shallow water equations. Modern total variation diminishing (TVD) and essentially non-oscillatory (ENO) methods have also been applied for the computation of free surface flows [2,3]. In each case, the flow equations were transformed from Cartesian to curvilinear coordinates, for applications on boundary-fitted meshes, and discretised using finite differences.

However, advantage may be gained by recasting the differential equations in integral form and employing a finite volume approach. Here, the solution variables are stored at cell centres and represent integral averages over each cell volume. Finite volume methods [4–12] combine the simplicity of finite difference methods with the geometric flexibility of finite element methods. This is because the dependent variables remain at all times referenced to the Cartesian frame even when using a curvilinear coordinate boundary-fitted mesh. Cell volumes and cell side vectors take the place of the transformation Jacobian and metric elements. In contrast, finite difference methods applied on curvilinear meshes inevitably lead to more complicated equation sets in the transformed coordinates. A number of finite volume methods which use Riemann based upwind schemes for the shallow water equations have been presented recently [8,13,10–12]. Despite the advantages of finite volume methods, the generation of a suitable boundary-fitted mesh remains, in general terms, non-trivial, particularly for complex multi-component flow geometries such as a channel, or estuary, with islands or bridge piers. One alternative is to use a finite element method which offers high geometric flexibility, particularly when implemented on an adaptive unstructured mesh. However, if the flow problem involves moving physical boundaries where the domain geometry changes then inevitably, local, and possibly global, re-meshing will be required several times during the calculation for as long as the boundary continues in motion. This can be time-consuming and the necessary interpolation of data between meshes may reduce the formal accuracy of the solution method. Such a scenario might arise in the case of a landslide [14]. Cases involving tidal storm surge in which parts of the domain become subject to flooding and drying are simpler to handle and do not require a moving boundary capability. These may be treated by pre-defining cells that may be fully or partially flooded during the calculation.

In this paper, we describe a solution method that potentially addresses all of the above mentioned difficulties. This is a high resolution finite volume scheme implemented on a Cartesian cut cell mesh that is applicable to problems involving complicated boundary contours with either static or moving bound-

aries. Cartesian cut cell methods were first developed within the aerospace community to provide a boundary-fitted mesh for multi-component flow geometries [15–19]. However, the basic principles are straightforward and have much wider application. Boundary contours are cut out of a background Cartesian mesh and cells that are partially or completely cut are singled out for special treatment. The remainder of the flow cells are treated in a straightforward manner. The cut cell method fully retains the boundary-conforming properties of a curvilinear mesh finite volume method or finite element method. However, there is no mesh generation in the conventional sense. This is replaced by relatively straightforward calculations for the boundary segment intersections with a background Cartesian mesh. The majority of the flow domain is overlaid with a regular Cartesian mesh so loss of solution accuracy due to pathological cases involving excessively stretched or skewed cells are avoided. In areas where the mesh resolution is too coarse to accurately resolve local geometric features, mesh adaptation is used. Moving flow boundaries are accommodated by recomputing local cell-boundary intersections as the boundaries move across a stationary background mesh, rather than re-meshing the whole flow domain, or large portions of it, as is necessary with other methods. This is particularly advantageous in cases where the physical geometry changes due to moving boundaries, as in the case of the landslide events considered later in the paper. Furthermore, the amplitude of boundary motion is unrestricted. An assessment of the accuracy of Cartesian cut cell approaches has been made recently by Coirier and Powell [20]. This analysis showed that the handling of complex irregular boundaries with cut cells did not reduce the global accuracy of the solution algorithm.

In order to develop a practical computational tool, two major issues need to be addressed. The first of these concerns the treatment of shallow water flows with complicated bed topography and boundary configurations. The flow features can be resolved accurately with a multi-dimensional high resolution upwind scheme whilst complex bed topography and boundary configurations can normally be represented accurately with the cut cell approach and mesh adaptation. A detailed description of the cut cell method and adaptation algorithms has been given previously by the authors for problems with static boundaries [21]. Here, we focus on the second major issue, namely extensions to the cut cell methodology for shallow water problems with moving boundaries. There are many practical examples where such a solver is needed. For example, following a landslide event, material may plunge into a lake or reservoir producing large surface waves leading to flood water inundation, loss of life and significant property damage [14]. These landslide-induced water waves result from the sudden motion of a solid boundary. Other examples are tsunami events [22]. Researchers have tackled tsunami generation by underwater landslides with a variety of numerical methods incorporating many different assumptions. Iwasaki [23] and Verriere and Lenoir [24] used linear potential theory to simulate wave generation by moving the domain bound-

ary. Non-linear shallow water models have been developed by Harbitz [25], Imamura and Gica [26] and Jiang and Le Blond [27–29], in combination with various landslide models. Fully nonlinear field equation models have also been developed by Grilli and Watts [30]. To deal with moving boundaries in the cut cell flow solver a special cell merging technique and boundary condition treatment are needed. Details of the required modifications and validation of the new method are reported here. The paper is organised as follows: in the next section, the Cartesian cut cell technique is outlined. Further description of the principles of the cut cell method for static geometries can be found in [21]; here, we concentrate on extensions of the methodology to flow geometries with moving boundary elements. The two-dimensional shallow water equations are given for a general moving reference frame in Section 3. In section 4, a suitable finite volume integration scheme, based on a MUSCL approach [31] and HLL approximate Riemann solver, is presented. The cut cell method is evaluated by recourse to a variety of test problems in section 5. Finally, in section 6, some conclusions are drawn.

2 The Cartesian Cut Cell Method

The cut cell method [18,19] employs a uniform stationary background Cartesian mesh and boundary contours are simply cut out of it. If the flow domain is multiply-connected, i.e. there is more than one solid region, these regions are specified with additional sets of data points. The background Cartesian mesh is constructed first and then any solid regions are simply cut out of it. Boundary intersections with the background grid define cut cells which have one of their sides coincident with a boundary segment. Thus, the method produces a boundary conforming mesh without the necessity to make the boundary a coordinate surface. In fact, there is no grid generation in the conventional sense; all that is necessary is to calculate the intersections of a series of line segments with the background Cartesian mesh. A detailed description of the principles of the cut cell method have been given previously by the authors [21]. The method can be readily extended to three space dimensions, should this be necessary for future developments [32].

3 The Flow Solver on a Cut Cell Mesh

3.1 The Shallow Water Equations

The shallow water equations are a reduced form of the depth averaged Navier-Stokes equations which represent the conservation of mass and momentum.

In differential conservative form the equations are

$$\frac{\partial \vec{U}}{\partial t} + \frac{\partial \vec{F}}{\partial x} + \frac{\partial \vec{G}}{\partial y} = \vec{Q}_s + \frac{\partial \vec{F}_v}{\partial x} + \frac{\partial \vec{G}_v}{\partial y} \tag{1}$$

where

$$\vec{U} = \begin{bmatrix} \phi \\ \phi u \\ \phi v \end{bmatrix}, \vec{F} = \begin{bmatrix} \phi u \\ \phi u^2 + \frac{\phi^2}{2} \\ \phi uv \end{bmatrix}, \vec{G} = \begin{bmatrix} \phi v \\ \phi uv \\ \phi v^2 + \frac{\phi^2}{2} \end{bmatrix}$$

$$\vec{Q}_s = \vec{A} + \vec{B} + \vec{C} + \vec{D}$$

$$\vec{A} = \begin{bmatrix} 0 \\ fv\phi \\ -fu\phi \end{bmatrix}, \vec{B} = \begin{bmatrix} 0 \\ \frac{g}{\rho}\tau_{xw} \\ \frac{g}{\rho}\tau_{yw} \end{bmatrix}, \vec{C} = \begin{bmatrix} 0 \\ -\frac{g}{\rho}\tau_{xf} \\ -\frac{g}{\rho}\tau_{yf} \end{bmatrix}, \vec{D} = \begin{bmatrix} 0 \\ \phi g b_x \\ \phi g b_y \end{bmatrix},$$

$\vec{F}$ and $\vec{G}$ are the convective flux vectors, $\vec{A}$ is the Coriolis force term, $\vec{B}$ is the wind shear stress term, $\vec{C}$ is the bed shear stress term, $\vec{D}$ is the bed slope term and the viscous stress flux vectors $\vec{F}_v$ and $\vec{G}_v$ are

$$\vec{F}_v = \begin{bmatrix} 0 \\ \sigma_{xx}\frac{\phi}{\rho} \\ \tau_{yx}\frac{\phi}{\rho} \end{bmatrix} \vec{G}_v = \begin{bmatrix} 0 \\ \tau_{xy}\frac{\phi}{\rho} \\ \sigma_{yy}\frac{\phi}{\rho} \end{bmatrix}.$$

In the above, f is the coriolis force, g is the acceleration due to gravity, ρ is the water density, u, v are the velocities in the x and y directions respectively, ϕ is the geopotential ($= gh$, h is the water depth), τ_{xw}, τ_{yw} are the wind shear stresses and τ_{xf}, τ_{yf} the bed shear stresses in the x, y directions, σ_{xx}, σ_{yy}, τ_{xy}, τ_{yx} are the normal and shear stress terms respectively and b_x, b_y are the bed slopes (measured downwards) in the x, y directions.

The integral form of the equations written in a general moving reference frame is

$$\frac{\partial}{\partial t}\int_{A_t} \vec{U}\ \mathrm{d}A + \oint_{S_t} \vec{H}.\vec{n}\ \mathrm{d}S = \int_{A_t} \vec{Q}_s\ \mathrm{d}A + \oint_{S_t} \vec{H}_v.\vec{n}\ \mathrm{d}S \tag{2}$$

where $\vec{H} =< \vec{F}, \vec{G} >$, $\vec{H}_v =< \vec{F}_v, \vec{G}_v >$ and $\mathbf{n}$ is the outward unit vector normal to the boundary S_t, which encloses the time-dependent area A_t.

In the present work, the viscous transport and source terms, $\vec{Q}_s$, are ignored

and we solve the convective flow equations

$$\frac{\partial}{\partial t}\int\limits_{A_t} \vec{U}\,\mathrm{d}A + \oint\limits_{S_t} \vec{H}\cdot\vec{n}\,\mathrm{d}S = \vec{0} \tag{3}$$

where

$$\vec{H} = \begin{pmatrix} \phi\vec{V} \\ \phi u\vec{V} + \frac{1}{2}\phi^2\vec{i} \\ \phi v\vec{V} + \frac{1}{2}\phi^2\vec{j} \end{pmatrix}. \tag{4}$$

$\vec{i}$ and $\vec{j}$ are the Cartesian unit base vectors and the contravariant velocity vector $\vec{V}$ is defined by

$$\vec{V} = \vec{v} - \vec{v}_s \tag{5}$$

where $\vec{v}$ and $\vec{v}_s$ are flow velocity vector and the velocity of the boundary of the control volume A_t, respectively. In our method, the background Cartesian mesh always remains stationary, even in the case of a moving boundary problem, therefore, $\vec{v}_s = 0$ at each cell interface; but at a moving solid boundary, $\vec{v}_s$ is the velocity of that boundary.

Although we have neglected the source terms in this paper, details of how these are discretised in the present finite volume method may be found in [33]. The required changes for a cut cell version of the method are relatively straightforward and will be reported in a future publication.

3.2 Finite Volume Discretisation of the Flow Equations

Since the inviscid form of the shallow water equations (3) provide a genuinely hyperbolic system, recent developments in numerical schemes for systems of conservation laws can be applied to them. Here, we use a proven MUSCL-Hancock finite volume scheme [11] with appropriate modifications for implementation on cut cell meshes with moving boundaries.

This is a high resolution, multi-dimensional Godunov-type scheme. Time integration is performed using a predictor-corrector scheme due to Hancock [34]. The MUSCL-Hancock scheme is second order accurate in both time and space in smooth regions. The predictor step uses a non-conservative approach, which

defines an intermediate value over a time interval $\Delta t/2$,

$$\left(A\vec{U}\right)_{ij}^{n+\frac{1}{2}} = \left(A\vec{U}\right)_{ij}^{n} - \frac{\Delta t}{2}\sum_{k=1}^{m}\vec{H}\left(\vec{U}_k\right)\cdot\vec{S}_k^n \tag{6}$$

where A is cell area, $\vec{S}_k$ is the side vector corresponding to cell face k defined as the unit normal vector multiplied by the length of cell face k, and m is maximum number of cell faces. For a flow (or uncut) cell, $m = 4$; for a cut cell, $m = 3$ to 5.

The flux vector function $\vec{H}(\vec{U}_k)$ is evaluated at the midpoints of cell faces following linear reconstruction of the flow solution within each cell using the stored cell centre data, via,

$$\vec{U}_k = \vec{U}_{ij}^n + \vec{r}_k\cdot\nabla\vec{U}_{ij}^n \tag{7}$$

where $\vec{r}_k$ is the normal distance vector from the cell centroid to face k and $\nabla\vec{U}_{ij}^n$ is a limited gradient vector in space (to be defined). The cell interface data is slope limited to avoid the creation of nonphysical overshoots or undershoots, using a MUSCL reconstruction procedure [35].

The gradients can be calculated trivially on flow cells away from the boundaries of the body. However, cells near solid boundaries, especially cut cells on solid boundaries, require special attention. Since a cut cell may have several fluid faces and a solid face, solid boundary conditions must be incorporated into the gradient calculation. Details of the cut cell gradient calculation are given in section 3.3.

The corrector step of the scheme is fully conservative. The intermediate solution from the predictor step is used to define a set of left and right-hand states for a series of Riemann problems at each cell interface. The solution of these Riemann problems provide a set of upwind interface fluxes which are used to update the flow solution over the time interval Δt, that is,

$$\left(A\vec{U}\right)_{ij}^{n+1} = \left(A\vec{U}\right)_{ij}^{n} - \Delta t\sum_{k=1}^{m}\vec{H}\left(\vec{U}_k^L,\vec{U}_k^R\right)\cdot\vec{S}_k^{n+\frac{1}{2}} \tag{8}$$

where the upwind flux $\vec{F}\left(\vec{U}_k^L,\vec{U}_k^R\right)$ is obtained by solving a local Riemann problem normal to the cell interface. The left and right-hand states at the interface k may be calculated by

$$\begin{aligned}\vec{U}_k^L &= \vec{U}_{ij}^{n+\frac{1}{2}} + \vec{r}_k^L\cdot\nabla\vec{U}_{ij}^n,\\ \vec{U}_k^R &= \vec{U}_{lm}^{n+\frac{1}{2}} + \vec{r}_k^R\cdot\nabla\vec{U}_{lm}^n\end{aligned} \tag{9}$$

where l and m relate to the right neighbouring cell.

To solve the Riemann problem, either an exact Riemann solver or various approximate Riemann solvers can be used; however, approximate solutions are much less computationally expensive. Here, the approximate Riemann solver of Harten, Lax and van Leer (HLL) [36] is used for the fluxes at the fluid interfaces of a cell because it is computationally efficient and robust in practice. For a cut cell interface coincident with a static or moving solid boundary, a different approach is used, based on an exact Riemann solution for a moving piston.

The time step employed is:

$$\Delta t = \nu \min(\Delta t_x, \Delta t_y) \tag{10}$$

where

$$\Delta t_x = \min_i \frac{A_{ij}}{\left|\vec{q}_{ij}.\vec{S}_{i+\frac{1}{2}j}\right| + \sqrt{\phi_{ij}}\left|\vec{S}_{i+\frac{1}{2}j}\right|} \tag{11}$$

with an analogous definition for Δt_y. The Courant number ν was taken in our calculations to be 0.5 which is close to the stability bound based on a linear von Neumann analysis.

3.2.1 The Riemann Solution at a Moving Boundary

The standard Riemann solver employed at cell interfaces (see [21]) must be modified to deal with moving boundaries. Consider a cut cell with a face coincident with a moving boundary, as shown in Figure 1, where $\vec{v}_f$ is the reconstructed flow velocity at the solid boundary and $\vec{v}_s$ is the solid boundary velocity. Projecting both $\vec{v}_f$ and $\vec{v}_s$ in directions normal to and tangential to the solid boundary with unit normal and unit tangential vectors $\vec{n}$, $\vec{t}$, we have

$$u_{fn} = \vec{v}_f \cdot \vec{n}, \quad u_{ft} = \vec{v}_f \cdot \vec{t}; \tag{12}$$

and

$$u_{sn} = \vec{v}_s \cdot \vec{n}, \quad u_{st} = \vec{v}_s \cdot \vec{t}. \tag{13}$$

In the tangential direction, any difference between u_{ft} and u_{st} is treated as a shear wave superimposed on a contact surface. In the normal direction the

Riemann solution for a moving piston is incorporated; hence, the solution for u_n^* can be found immediately, that is

$$u_n^* = u_{sn} \tag{14}$$

and ϕ_n^* can be obtained by comparing the two velocities u_{fn} and u_{sn}.

Thus, if $u_{fn} < u_{sn}$, both left and right moving waves are rarefactions. Then ϕ_n^* can be obtained from the rarefaction relations [37],

$$\phi_n^* = \frac{1}{4}\Big(u_{fn} - u_n^* + 2\sqrt{\phi_f}\Big)^2. \tag{15}$$

If, on the other hand, $u_{fn} \geq u_{sn}$, both left and right moving waves are bores and ϕ_n^* is obtained from the Rankine-Hugoniot conditions,

$$(\phi_n^* - \phi_f)\sqrt{\frac{\phi_n^* + \phi_f}{2\phi_n^*\phi_f}} + u_n^* - u_{fn} = 0. \tag{16}$$

Although there is no closed form solution for (16), ϕ_n^* can be obtained to the required accuracy by iteration. However, an approximate solution to (16) can be found as follows. First, we put

$$F\left(\phi_n^*\right) = (\phi_n^* - \phi_f)\sqrt{\frac{\phi_n^* + \phi_f}{2\phi_n^*\phi_f}} + u_n^* - u_{fn}, \tag{17}$$

and then expand (17) in a Taylor series about the point ϕ_f and retain first order terms,

$$F\left(\phi_n^*\right) \approx F\left(\phi_f\right) + F'\left(\phi_f\right)\left(\phi_n^* - \phi_f\right) = 0 \tag{18}$$

where

$$F\left(\phi_f\right) = \left.F\left(\phi\right)\right|_{\phi\,=\,\phi_f} = u_n^* - u_{fn}, \tag{19}$$

$$F'(\phi_f) = \left.\frac{dF}{d\phi}\right|_{\phi\,=\,\phi_f} = \frac{1}{\sqrt{\phi_f}}. \tag{20}$$

Hence, from (18) we have

$$\phi_n^* \approx \phi_f - \frac{F\left(\phi_f\right)}{F'\left(\phi_f\right)}. \tag{21}$$

Once ϕ_n^* is known, the required flux function normal to the moving boundary is

$$\vec{F}_n^* = \begin{bmatrix} \phi_n^* \left(u_n^* - u_{sn}\right) \\ \phi_n^* u_n^* \left(u_n^* - u_{sn}\right) + \frac{1}{2}{\phi_n^*}^2 \\ \phi_n^* v_n^* \left(u_n^* - u_{sn}\right) \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{2}{\phi_n^*}^2 \\ 0 \end{bmatrix} \tag{22}$$

and the required flux side vector function at the moving cut cell face is

$$\vec{H}^* \cdot \vec{S} = \begin{bmatrix} 0 \\ \frac{1}{2}{\phi_n^*}^2 \Delta y \\ -\frac{1}{2}{\phi_n^*}^2 \Delta x \end{bmatrix} \tag{23}$$

where $\vec{S}$ (with components Δx, Δy) is the side vector corresponding to the moving solid face.

3.3 Calculation of Gradients and Reconstruction Technique on Moving Boundaries

Second order accuracy in space can be achieved by reconstructing the flow data within each cell based on the stored data at cell centres and appropriate gradient information. The gradient calculation is straightforward on uncut flow cells located away from a solid boundary (see, [11]). For cells cut by a solid boundary, however, a different gradient calculation is needed. The relevant boundary conditions in the gradient calculation must also accommodate both static and moving boundaries.

Here, reflection boundary conditions are used at a solid boundary and the required variables in fictional cell R are obtained as follows (see Figure 2)

$$\begin{cases} \phi_R = \phi_{ij} \\ \vec{v}_R = \vec{v}_{ij} - 2\left(\vec{v}_{ij} \cdot \vec{n}\right)\vec{n} + 2\left(\vec{v}_s \cdot \vec{n}\right)\vec{n}. \end{cases} \tag{24}$$

The gradients on cut cell (i, j) may be of two types: fluid and solid. We calculate the fluid gradients and solid gradients separately, that is,

$$\vec{U}_x^f = G\left(\frac{\vec{U}_{i+1,j} - \vec{U}_{i,j}}{\Delta x_{i+\frac{1}{2},j}}, \frac{\vec{U}_{i,j} - \vec{U}_{i-1,j}}{\Delta x_{i-\frac{1}{2},j}}\right) \tag{25}$$

$$\vec{U}_y^f = G\left(\frac{\vec{U}_{i,j+1} - \vec{U}_{i,j}}{\Delta y_{i,j+\frac{1}{2}}}, \frac{\vec{U}_{i,j} - \vec{U}_{i,j-1}}{\Delta y_{i,j-\frac{1}{2}}}\right) \tag{26}$$

where $\Delta x_{i+\frac{1}{2},j} = x_{i+1,j} - x_{i,j}$, $\Delta y_{i,j+\frac{1}{2}} = y_{i,j+1} - y_{i,j}$ and

$$\vec{U}_x^s = G\left(\frac{\vec{U}_R - \vec{U}_{i,j}}{\Delta x_{i,R}}, \frac{\vec{U}_{i,j} - \vec{U}_{i-1,j}}{\Delta x_{i-\frac{1}{2},j}}\right) \tag{27}$$

$$\vec{U}_y^s = G\left(\frac{\vec{U}_{i,j+1} - \vec{U}_{i,j}}{\Delta y_{i,j+\frac{1}{2}}}, \frac{\vec{U}_{i,j} - \vec{U}_R}{\Delta y_{j,R}}\right) \tag{28}$$

where $\Delta x_{i,R} = x_R - x_{i,j}$, $\Delta y_{j,R} = y_{i,j} - y_R$ and G is a slope limiter function that is used to prevent over- or under-shoots. In the present calculations the k limiter,

$$G(a,b) = s \cdot \max[0, \min(k|b|, s \cdot a), \min(|b|, ks \cdot a)], \tag{29}$$

where $s = \text{sign}(b)$ and $1 \leq k \leq 2$. Differences in the choice of k are marginal with this scheme and do not affect its numerical stability. In practice, $k = 1.5$ has been found to give the best performance.

A length average technique is then used to obtain unique gradients in the cut cell,

$$\vec{U}_x = \frac{\Delta y_s \vec{U}_x^s + \Delta y_f \vec{U}_x^f}{\Delta y} \tag{30}$$

$$\vec{U}_y = \frac{\Delta x_s \vec{U}_y^s + \Delta x_f \vec{U}_y^f}{\Delta x} \tag{31}$$

where $\Delta x_f = |AB|, \Delta x_s = |BC|, \Delta y_s = |CD|$ and $\Delta y_f = |DE|$. Δx and Δy are the uncut cell side lengths in the x and y directions, respectively.

Since $\Delta x_f + \Delta x_s = \Delta x$ and $\Delta y_f + \Delta y_s = \Delta y$, we note that if $\Delta y_f = \Delta y$, $\Delta y_s = 0$, so $\vec{U}_x = \vec{U}_x^f$; otherwise, if $\Delta y_s = \Delta y$, $\Delta y_f = 0$, so $\vec{U}_x = \vec{U}_x^s$. $\vec{U}_x$ and $\vec{U}_y$ are components of a gradient vector in the cut cell, that is

$$\nabla \vec{U}_{ij} = \begin{bmatrix} \vec{U}_x \\ \vec{U}_y \end{bmatrix}. \tag{32}$$

Given the gradient vector $\nabla\vec{U}_{ij}$, a reconstructed solution vector $\vec{U}(x,y)$ can be found anywhere within the cut cell from

$$\vec{U}(x,y) = \vec{U}_{ij} + \vec{r}\cdot\nabla\vec{U}_{ij} \tag{33}$$

where $\vec{r}$ is the normal distance vector from the cell centroid to any specific interface or solid boundary.

3.4 Cell Merging at Moving Boundaries

As described in section 2, a cut cell mesh is formed by cutting solid boundaries out of a background Cartesian mesh. Thus, in practice, arbitrarily small cells may of course be created near solid boundaries. This gives rise to an unrealistically small time step locally since the MUSCL-Hancock scheme is conditionally stable. Similar problems may arise with moving boundaries as they move across the stationary background mesh. When boundaries move relative to the background mesh, cells near the moving boundaries undergo changes. In general, these changes fall into three categories: (1) **cut cell becomes solid cell**; (2) **cut cell remains unchanged**; (3) **flow cell becomes Cut cell**. Categories two and three cause no problems. However, category one, where a cut cell becomes solid corresponds to a zero cell volume and this would cause problems within the finite volume solver unless modifications are made.

Fortunately, these problems can be solved very simply by a cell merging technique. The basic idea is to combine several neighbouring cells together so that any interfaces between merged cells are ignored and waves can travel in a newly combined larger cell without reducing the global time step. This approach has been used previously by the present authors for static boundary cases [21]. Here, we focus on adjustments to the flow solver needed at moving boundaries.

See Figure 3, suppose the time step Δt is based on flow cell B. This will cause two problems at cut cell A: (1) one of numerical stability due to the smaller cell size; (2) cut cell A becomes solid after the time interval Δt. To implement cell merging, we first consider the standard unmodified updates at cells A and B and illustrate what must be done by reference to the predictor step of the MUSCL-Hancock solver (6); changes to the corrector step (8) follow analogously. Thus,

$$\Delta(A\vec{U})_A = -\frac{\Delta t}{2}\sum_{k=1}^{m_A}\vec{H}_k\cdot\vec{S}_k, \tag{34}$$

$$\Delta(A\vec{U})_B = -\frac{\Delta t}{2}\sum_{k=1}^{m_B}\vec{H}_k\cdot\vec{S}_k \tag{35}$$

Then, we ignore the interface between cell A and B, and update the merged cell C simply by combining the volume updates of cells A and B,

$$\Delta(A\vec{U})_C = \Delta(V\vec{U})_A + \Delta(V\vec{U})_B \tag{36}$$

or

$$\Delta(A\vec{U})_C = -\frac{\Delta t}{2}\left(\sum_{k=1}^{m_A}\vec{H}_k\cdot\vec{S}_k + \sum_{k=1}^{m_B}\vec{H}_k\cdot\vec{S}_k\right) \tag{37}$$

It should be noted that the fluxes on the interface $|cd|$ between cell A and B cancel out automatically since the flux calculation is conservative. In the modified scheme, the flux balance is now being carried out over each side of the enlarged cell C; this cell will generally have more than four sides. The conserved variable $\vec{U}$ for cell C at time t^{n+1} is

$$A_C^{n+1}\vec{U}_C^{n+1} = A_A^n\vec{U}_A^n + A_B^n\vec{U}_B^n - \frac{\Delta t}{2}\left[\sum_{k=1}^{m_A}\vec{H}_k^n\cdot\vec{S}_k^n + \sum_{k=1}^{m_B}\vec{H}_k^n\cdot\vec{S}_k^n\right] \tag{38}$$

Although cut cell A finally disappears, its contribution to the conserved dependent variables will be transfered into neighbouring cells so that strict conservation is maintained automatically. In principle, cell merging may reduce integration accuracy at solid boundaries but in practice it has been shown not to adversely affect the global accuracy of the calculation method [20].

4 Quad-tree Mesh Adaptation

The present cut cell method based on a uniform background grid will of course not provide sufficient flexibility for practical applications. For example, to represent highly irregular coastline it would be necessary to employ a globally fine mesh. Clearly, this would neither be computationally efficient nor practically feasible. Cell sizes varying by orders of magnitude between the smallest and largest are desirable in practice. In common with other techniques, such as a finite element methods, this flexibility can be met by employing mesh adaptation. Within the present cut cell method, the process can be largely automated based on some simple rules [21].

The practical potential of the adaptive cut cell technique is illustrated by an application to the upper reaches of the Manchester Ship Canal, UK (for other

applications see [21]). Figure 4a shows the result of meshing the whole region using square background cells with $\Delta x = \Delta y = 20$m and performing refinement to boundaries; the smallest grid cells in the mesh are around 0.30m^2. Figure 4b-c show enhanced detail of the local mesh around the periphery of the domain. It can be seen that the method copes well with a narrow meandering flow domain providing fine mesh resolution compatible with the degree of irregularity in the boundary contours and local bathymetry. The end result is a high resolution boundary fitted mesh with a spatial resolution comparable to that of an adaptive finite element method. A feature of the data structures employed is that cells may be created and removed very easily. Consequently, dynamic adaptation to moving boundaries and/or evolving flow features during run time is also possible in principle although these issues are not discussed further here. The purpose is to show that the cut cell method can provide adequate resolution of complex flow domains and that the method is well founded for applications to moving boundaries. Extensions to include dynamic mesh adaptation will be the subject of a future publication.

5 Numerical Results and Discussion

In the following calculations, two types of boundary conditions are encountered: those corresponding to solid and transmissive boundaries. No flow is permitted through a solid boundary and therefore, at a solid interface, $q.\vec{S} = 0$ which means that the corresponding flux vector $\vec{H}.S$ depends only on the geopotential. The required value for the geopotential is taken from the adjacent fluid cell. At a transmissive boundary, the data necessary to compute the flux vector $\vec{H}$ at the interface is taken to be that at the adjacent interior cell centre.

Examples that show the efficiency of the basic cut cell method for static boundaries can be found in [21].

5.1 Supercritical Flow Past a Ship's Hull

To validate the present cut cell method for moving boundary problems the flow field round a stylised ship's hull travelling at a supercritical velocity ($Fr = 3$) has been simulated using both a hull moving through quiescent water and a static hull placed in moving water. In principle, an identical flow field should be expected in each case. The numerical experiments have been conducted in a flume 20m long and 6m wide using a hull 7m long and 2m across, having a bow 4m long and a square stern. Since the computations have been undertaken using the shallow water equations, the hull is considered to extend to the bed

of the flume and to have sufficient freeboard to prevent it being over-topped. Both cases were started impulsively and the evolved flow fields were compared at $t = 2$s. The quiescent water depth in the flume was 0.4m which requires the hull to be moving with a velocity of 5.9ms^{-1} to achieve the required Froude number. In the case where the hull was moving it was started with the stern in contact with the downstream end of the flume (i.e. at $x = 0$), whilst in the case where the hull was held stationary in a stream flowing right to left, the stern was located at $x = 11.9$m so as to ensure co-location at $t = 2$s.

In the numerical computations with the benefit of symmetry about the longitudinal axis half the flow field has been calculated using a mesh with 400×60 nodes and $\Delta x = \Delta y = 0.05$m. The upstream and downstream boundaries were taken to be transmissive boundaries whilst the symmetry plane and the flume wall were modelled using the solid (flow tangency) boundary condition. Table 1 shows the required number of time steps and CPU times for the two calculations which were performed on an Alpha-based workstation with a 600MHz processor running Red-Hat Linux 6.0 and using the Digital Fortran compiler with full optimisation.

Figure 5 shows the simulated water surface at $t = 2.0$s. The water surface on the port side of the hull corresponds to the moving case whilst that on the starboard side corresponds to the static case. The flow solutions in the vicinity of and immediately downstream of the hull are indistinguishable. The only differences in the flow fields are observable near the downstream (right of figure) exit of the flume where a start-up vortex has formed in the moving case while, of course, no vortex exists in the static case. These differences due simply to transients caused by the impulsive startup should be ignored. The letters on Figure 5 are the reference points for the water depth data given in Table 2. Using oblique shock wave theory [38] analytical values for the water depth at locations A, B, C, D and E can be calculated and these are also given.

Figure 6 shows line contours of water depth at $t = 2.0$s for the two cases. The upper half of Figure 6 shows the results for the moving hull and the lower half for the static hull. The contour plots for the two cases are identical until just downstream of the point where the bow wave, reflected from the side walls of the flume, is reflected at the centreline. Downstream of this point slight asymmetries are observed due to the start-up vortex in the moving hull case.

These results provide confirmation of the validity of the cut cell method for problems with moving boundary elements.

5.2 A Hypothetical Landslide

A hypothetical landslide problem has been calculated in order to demonstrate the potential applicability of the method to such problems. The problem consists of a rectangular lake 200m×150m which has a uniform depth of 10m. The water in the lake is quiescent. The landslide, with a front 40m across, enters the lake in the centre of the southern side travelling at a speed of 14.8ms^{-1}. This speed corresponds to a Froude number of around 1.5. The landslide has been simulated using a solid obstruction with a slightly pointed, symmetrical, front face ($\theta = 10°$).

In the numerical computations the full transient flow field has been calculated using two grids: a fine mesh with 400×300 nodes and $\Delta x = \Delta y = 0.5$m, and a coarse mesh with 200×150 nodes and $\Delta x = \Delta y = 1.0$m. In both cases, the origin of co-ordinates is taken as the southwest corner of the lake. The solution is allowed to evolve for five seconds by which time the landslide has intruded a lateral distance of 74m into the lake. At the boundaries of the lake, solid (flow tangency) boundary conditions are applied. The solid body representing the landslide mass is taken as extending vertically upwards from the bed and having sufficient freeboard to ensure it is not over-topped by the bow wave created by its motion. Based on the calculations carried out on the two grids grid convergence index (GCI) data [21] has been calculated in order to estimate the local percentage errors in the calculation. Table 3 shows the error estimates at a number of points in the lake lying within the disturbance region. The data indicate that the calculations on the fine mesh have been obtained to an accuracy of half of one per-cent or better. The two calculations were performed using a Alpha based workstation with a 600MHz processor running Red-Hat Linux 6.0 and using the Digital Fortran compiler with full optimisation.

Figure 7 shows the simulated water surface for the fine grid calculation at $t = 5.0$s. By this time the bow wave is well developed and the shock standoff distance (between the leading edge of the landslide mass and the bow wave) is constant (at approximately 28.5m). The bow wave is highly curved and weakens significantly, losing amplitude with distance from the landslide. The water rises about 11m from its quiescent level through the bow wave. Behind the bow wave, the water level continues to rise slowly reaching its maximum height on the front face of the landslide mass (approx 12.5m above the quiescent level). A strong centred depression forms at the front corners of the landslide mass as the displaced water expands around the corner causing the water level to drop rapidly. Because of the strength of the centred depression the water level drops lower than its undisturbed equilibrium level and, further round the side of the landslide mass, a hydraulic jump forms to recover the equilibrium state.

Figure 8 shows line contours of the water depth at $t = 5.0$s for the two grids (at approximately 0.5m intervals). The results for both cases are almost identical except for a slight thickening of the bore waves on the coarse grid. In both cases the the density of the contour lines confirms the strength of the two centred depressions at the front edges of the landslide mass. The data points shown on the contours were used to calculate the GCI values presented in Table 3 which represent the percentage error in the numerical solution at the specified location.

5.3 Landslide in an Fjord

Rock-falls and landslides into lakes and fjords are known to generate extreme waves which propagate at high speed and can cause significant damage [14]. Such a simulation is presented here. A hypothetical fjord 4 km long, 500m wide and 30m deep (see Figure 9) has been discretised using a uniform mesh with $\Delta x = \Delta y = 25$m. The coastline is modelled using Cartesian cut cells thus making the assumption that the coastline is formed by semi-infinite vertical cliffs. For simplicity, the effects of bed slope and varying bathymetry have been neglected in the model. These simplifying assumptions are not unwarranted as glaciated valleys and fjords are typified by a U shaped cross section, with sheer sides and a flat bottom. At the landward end the fjord divides in two forming northern and southern arms approximately 1 km long. A landslide lasting six seconds has been generated along a 450m frontage in the southern arm, and the following three minutes have been simulated.

Noda [39] suggests that landslides can be modelled using a moving vertical wall, whose velocity, V, is related to the maximum height of the wave, η_{max}

$$\eta_{max} = 1.32 \frac{dV}{\sqrt{gd}}. \tag{39}$$

Thus, for a landslide travelling at 34ms^{-1} (Froude number, $Fr = 2$) the maximum wave height is 79.2m. Figure 10 shows the predicted wave height distribution in front of a wall moving at the same speed in water 30m deep. The simulations predict a wave 57m tall will be formed; this agrees closely with the height predicted by the moving piston Riemann solver (section 3.2.1). A wall velocity of 34ms^{-1} is about half of that reported for Lituya Bay, Disenchantment Bay and Lanfjord landslides ([14]).

In the present example, the landslide is simulated by moving the wall at 34ms^{-1} for 6s. The wall starts and stops instantaneously, generating a bore wave followed by a depression. The landslide generates an initial wave 57m in height. This height is dramatically increased as the wave interacts with the

southern wall of the fjord causing it to reach 100m in height (see Figure 11).

One minute after the start of the landslide (Figure 12) the main wave has travelled approximately 2km from the cliff and has been reduced in amplitude (to 20m high). The reduction in height is caused by energy loss during the diffraction of the wave into the northern arm. The diffracted wave has reflected from the northeastern wall of the fjord at this time and is propagating back towards the southern arm.

One minute later (Figure 13), the main wave has propagated 3.2km from the cliff face and has weakened considerably, now being approximately 10m high. The further loss in amplitude is caused by the combined effects of refraction and reflection in the main channel of the fjord. At this time the secondary wave generated by reflection is being focused in the southwestern corner of the southern arm. This reverberation effect will continue for some time.

Three minutes after the landslide began (Figure 14) the main wave has begun to diffract into the open sea. Finally, the distribution of water depth along the main channel at various times is shown in Figure 15. This is shown at the position indicated in Figure 9.

6 Conclusions

A new computational method has been presented for the calculation of shallow water flow problems with moving boundaries. The method is based on a Cartesian cut cell approach and multi-dimensional high resolution upwind scheme. By using cut cells on a background Cartesian mesh, solid regions are simply cut out of a background grid; hence the present method offers a fully boundary-fitted gridding capability which can cope with complex shallow water flows around arbitrarily complicated topography. No grid generation is required in the conventional sense. This is replaced by relatively simple calculations for the coordinates of the points at which boundary segments intersect the background grid. In cases with moving boundary elements, all that is necessary is to update the cut cell data locally for as long as a boundary element moves across the stationary background mesh. No moving or overlaid meshes are required. A high-order MUSCL-Hancock finite volume scheme used in conjunction with slope limiting and an HLL approximate Riemann solver provides high resolution solutions to the shallow water equations. This enables the method to deal with complex shallow water flows such as dam-breaks, bore waves and general subcritical or supercritical flows involving strong discontinuities. When used in conjunction with local mesh adaptation, the method provides high resolution of local geometric features such as irregular coastline and variable bathymetry. The method has been applied to the case of supercritical flow past a ship's hull

and to hypothetical landslide events. The results indicate the promise of the new method for solving a wide range of shallow water flows involving arbitrarily complicated configurations with static or moving boundaries. The method can be extended to very large domains, such as ocean basins, where spherical coordinates are employed. The grid and geometry would need to be specified using latitude and longitude with appropriate modifications to the equation set. In future work, the new cut cell method will be modified to incorporate dynamic mesh adaptation at run time in order to provide improved spatial resolution of key transient flow phenomena and to incorporate the source terms neglected in the present study.

References

[1] P. Glaister, Flux Difference Splitting for Open-Channel Flows, International Journal Numerical Methods Fluids 16 (1993) 629–654.

[2] J. Yang, C. Hsu, Computation of Free Surface Flows, Journal of Hydraulic Research 31 (3) (1993) 403–413.

[3] M. Nujić, Efficient Implementation of Non-Oscillatory Schemes for the Computation of Free Surface Flows, Journal of Hydraulic Research 33 (1) (1995) 101–111.

[4] R. MacCormack, A. Paullay, Computational Efficiency Achieved by Time Splitting of Finite Difference Opperators, Paper 72–154, AIAA (1972).

[5] D. Causon, Methods for suppressing oscillations arising in the numerical solution of hyperbolic partial differential equations with applications in transonic flow problems, Ph.D. thesis, University of Salford, UK (1979).

[6] D. Causon, P. Ford, Numerical Solution of the Euler Equations for Three Dimensional Flows, The Aeronautical Journal 89 (886) (1985) 226–241.

[7] C. Bellos, J. Soulis, J. Sakkas, Computation of Two Dimensional Dam Break Induced Flows, Advances in Water Resources 14 (1) (1991) 31–41.

[8] F. Alcrudo, P. Garcia-Navarro, A High-Resolution Godunov-Type Scheme in Finite Volumes for the 2D Shallow-Water Equations, International Journal of Numerical Methods in Fluids 16 (1993) 489–505.

[9] D. Zhao, H. Shen, G. Tabios, J. Lai, W. Tan, Finite Volume Two Dimensional Unsteady Flow Model for River Basins, Journal of Hydraulic Engineering 120 (7) (1994) 863–883.

[10] D. Zhao, H. Shen, J. Lai, G. Tabios, Approximate Riemann Solvers in FVM for 2D Hydraulic Shock Wave Modelling, Journal of Hydraulic Engineering 122 (12) (1996) 692–702.

[11] C. Mingham, D. Causon, A High Resolution Finite Volume Method for Shallow Water Flows, Journal of Hydraulic Engineering 124 (6) (1998) 605–614.

[12] C. Mingham, D. Causon, Calculation of Unsteady Bore Diffraction Using a High Resolution Finite Volume Method, Journal of Hydraulic Research 38 (1) (2000) 1–15.

[13] D. Ambrosi, Approximation of Shallow Water Equations by Roe's Riemann Solver, International Journal Numerical Methods Fluids 20 (1995) 157–168.

[14] B. Voight (Ed.), Rockslides and Avalanches, 2 : Engineering Sites, Vol. 14B of Developments in Geotechnical Engineering, Elsevier, Amsterdam, 1979.

[15] B. Wedan, J. South Jr, A Method for Solving the Transonic Full Potential Equation for General Configurations, Paper 83–1889, AIAA (1983).

[16] D. De Zeeuw, P. KG, An adaptively refine cartesian mesh solver for the euler equations, Journal of Computational Physics 104 (1) (1993) 56–68.

[17] M. Berger, M. Aftomis, J. Melton, Adaption and surface modelling for cartesian mesh methods, AIAA–95–1725–CP (1995) 881–891.

[18] G. Yang, D. Causon, D. Ingram, R. Saunders, P. Batten, A Cartesian Cut Cell Method for Compressible Flows - Part A : Static Body Problems, Aeronautical Journal 101 (1001) (1997) 47–56.

[19] G. Yang, D. Causon, D. Ingram, R. Saunders, P. Batten, A Cartesian Cut Cell Method for Compressible Flows - Part B : Moving Body Problems, Aeronautical Journal 101 (1001) (1997) 57–65.

[20] W. Coirier, K. Powell, An accuracy assessment of cartesian mesh approaches for the euler equations, Journal of Computational Physics 117 (1993) 121–131.

[21] D. Causon, D. Ingram, C. Mingham, G. Yang, R. Pearson, Calculation of shallow water flows using a cartesian cut cell approach, Advances in Water Resources 23 (2000) 545–562.

[22] Y. Kawata, Tsunami in Papua New Guinea was intense as first thought, Eos Trans. Am. Geophys. Union 80 (9) (1999) 101.

[23] S. Iwasaki, The wave forms and directivity of a tsunami generated by an earthquake and a landslide, Science of Tsunami Hazards 15 (1997) 23–40.

[24] M. Verriere, M. Lenoir, Computation of waves generated by submarine landslides, International Journal of Numerical Methods in Fluids 14 (1992) 403–421.

[25] C. Harbitz, Model simulations of tsunamis generated by the storegga slides, Journal of Marine Geology 105 (1992) 1–21.

[26] F. Imamura, E. Gica, Numerical model for tsunami generation due to subaqueous landslide along a coast, Science of Tsunami Hazards 14 (1996) 13–28.

[27] L. Jiang, P. Le Blond, The coupling of a submarine slide and the surface waves which it generates, Journal of Geophysical Research 97 (C8) (1992) 12731–12744.

[28] L. Jiang, P. Le Blond, Numerical modelling of an underwater bingham plastic mud slide and the waves which it generates, Journal of Geophysical Research 98 (C6) (1993) 10303–10317.

[29] L. Jiang, P. Le Blond, Three dimensional modelling of tsunami generation due to a submarine mud slide, J. Phys. Ocean. 24 (1994) 559–573.

[30] S. Grilli, P. Watts, Modelling of waves generated by a moving submerged body: Applications to underwater landslides, Engrg. Analysis Boundary Elements 23 (8) (1999) 645–656.

[31] B. van Leer, Towards the Ultimate Conservative Difference Scheme IV. A New Approach to Numerical Convection, Journal of Computational Physics 23 (1977) 276–299.

[32] G. Yang, D. Causon, D. Ingram, Calculation of compressible flows about complex moving geometries using a 3d Cartesian cut cell method, International Journal for Numerical Methods in Fluids 33 (2000) 1121–1151.

[33] K. Hu, C. Mingham, D. Causon, A bore-capturing finite volume method for open channel flows, International Journal for Numerical Methods in Fluids 28 (1998) 1241–1261.

[34] B. van Leer, On the Relation Between the Upwind-Differencing Schemes of Godunov, Engquist-Osher and Roe, SIAM Journal on Scientific and Statistical Computing 5 (1) (1984) 1–20.

[35] B. van Leer, Towards the Ultimate Conservative Difference Scheme V. A Second Order Sequel to Godunov's Method, Journal of Computational Physics 32 (1979) 101–136.

[36] A. Harten, P. Lax, B. van Leer, On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws, SIAM Review 25 (1) (1983) 35–61.

[37] J. Stoker, Water Waves: The Mathematical Theory with Applications, wiley classics library Edition, John Wiley & Sons, New York, 1992.

[38] D. Causon, C. Mingham, D. Ingram, Advances in calculation methods for supercritical flow in spillway channels, Journal of Hydraulic Engineering 125 (10) (1999) 1039–1050.

[39] E. Noda, Water waves generated by landslides, Proc. ASCE, Journal of Waterways and Harbours Division 96 (WW4) (1970) 835–855.

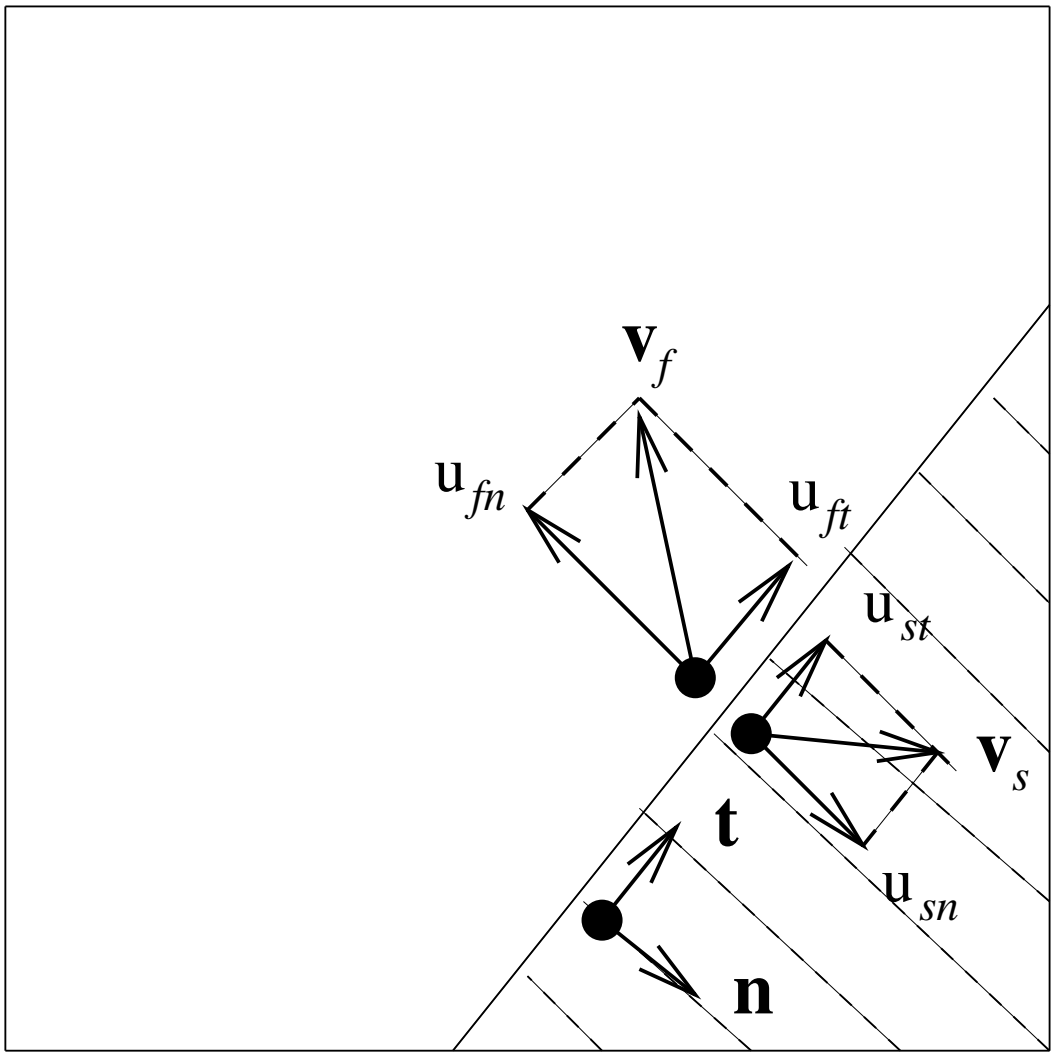


Fig. 1. A cut cell with a moving boundary

Table 1
Supercritical flow past a ship's hull: CPU times (seconds) for the two cases

case	total CPU	time steps	CPU per time step
Moving	95.14	687	0.138
Static	104.0	771	0.135

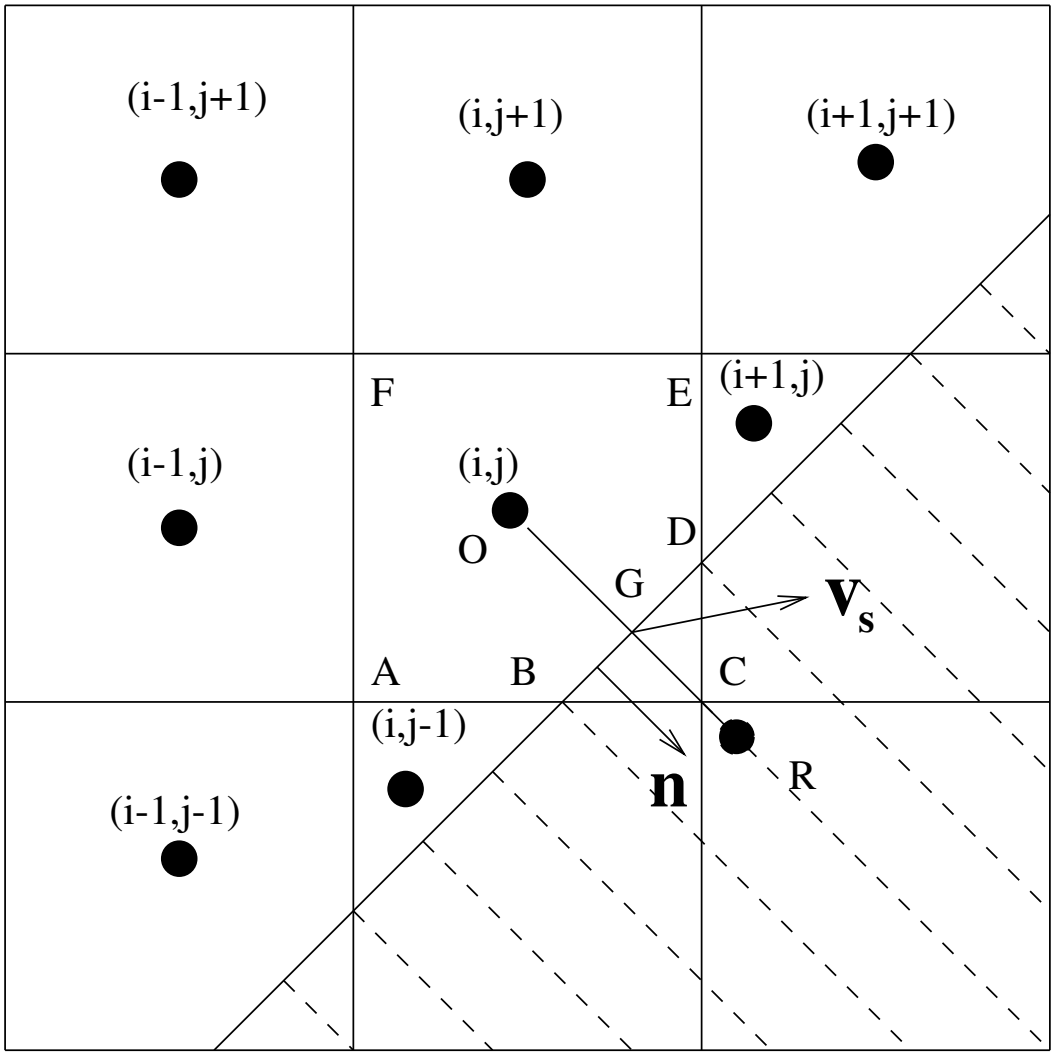
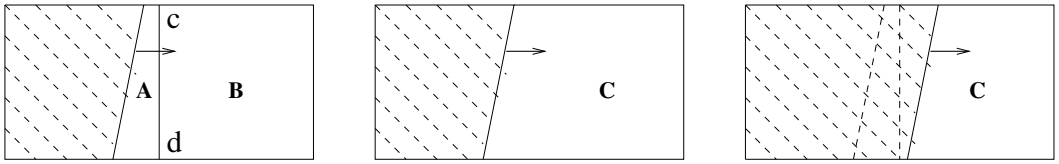


Fig. 2. Calculation of gradients on a cut cell



(a) at t^n (b) at t^n (c) at t^{n+1}

Fig. 3. Cell merging technique for a moving boundary.

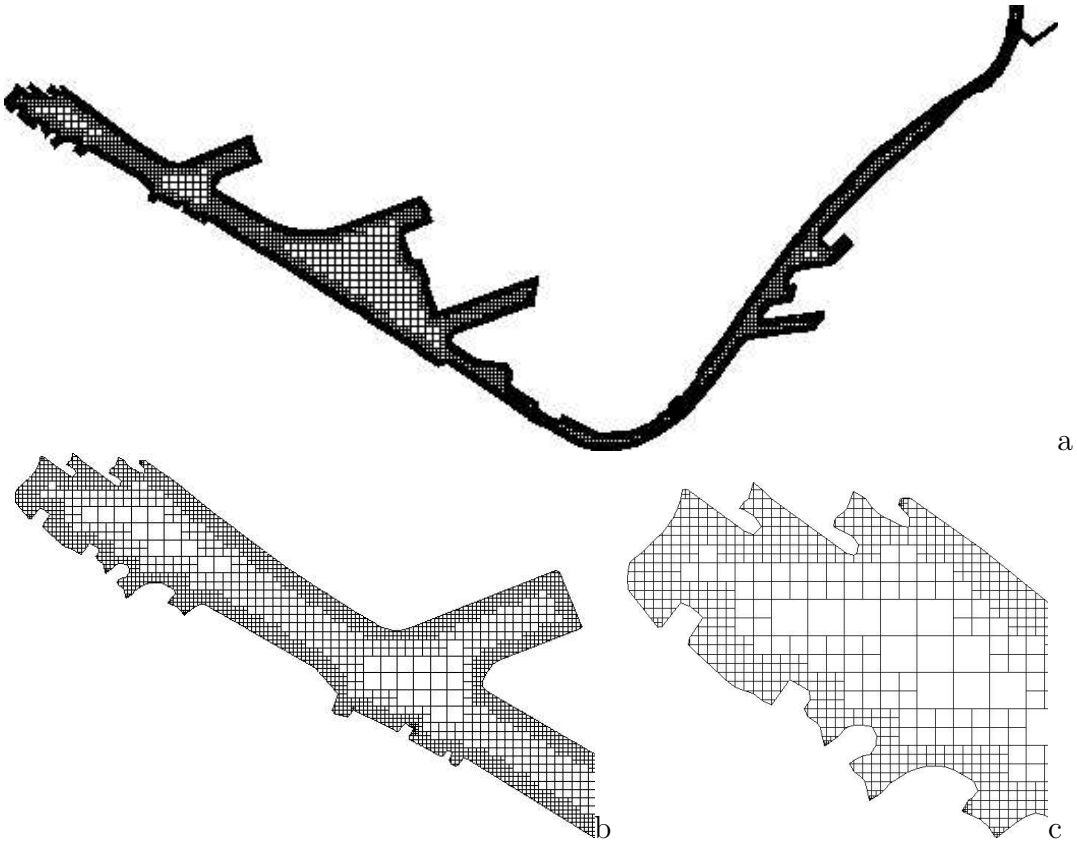


Fig. 4. Quadtree mesh for the upper reaches of the Manchester Ship Canal: a) general layout, b), and c) enhanced detail of local features.

Table 2
Supercritical flow past a ship's hull: Comparison of water depths (in metres) with analytical solutions for the two cases at selected points.

Point	Moving	Static	Analytical
A	0.40	0.40	0.40
B	0.75	0.75	0.75
C	0.38	0.38	0.38
D	0.00	0.00	0.00
E	1.21	1.21	1.21
F	0.23	0.23	—

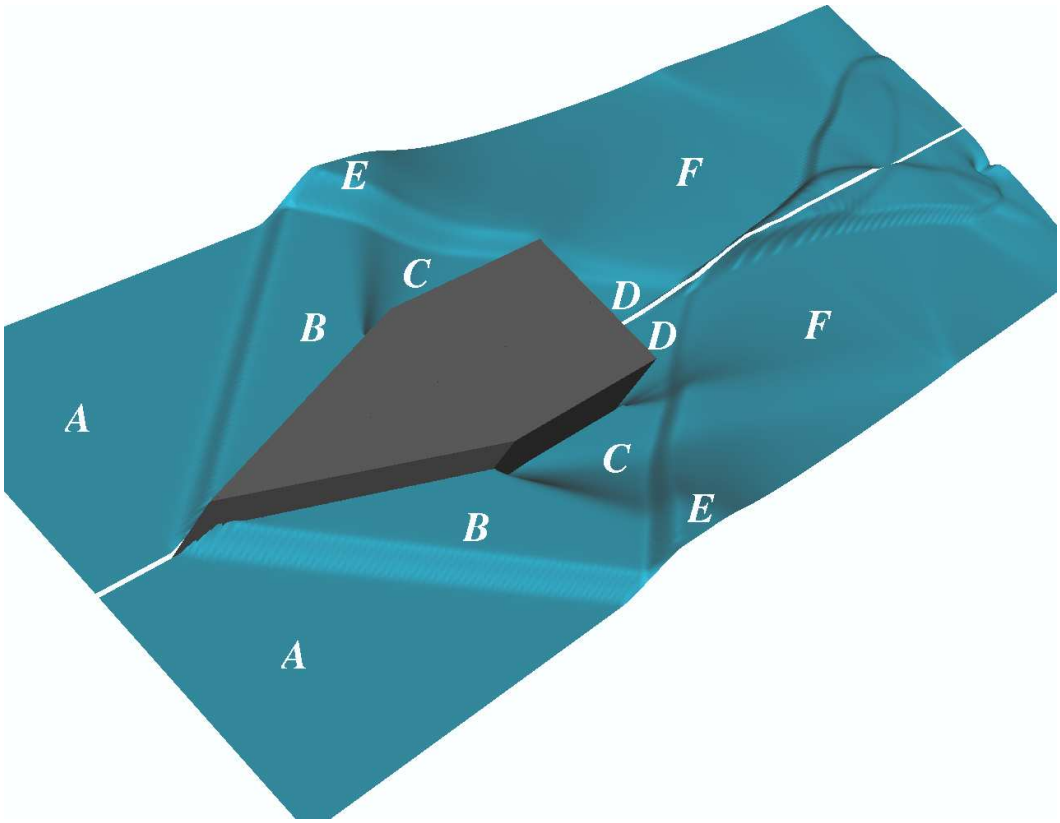


Fig. 5. Supercritical flow past a ship's hull: Simulated water surface for the moving (port side) and static (starboard side) cases.

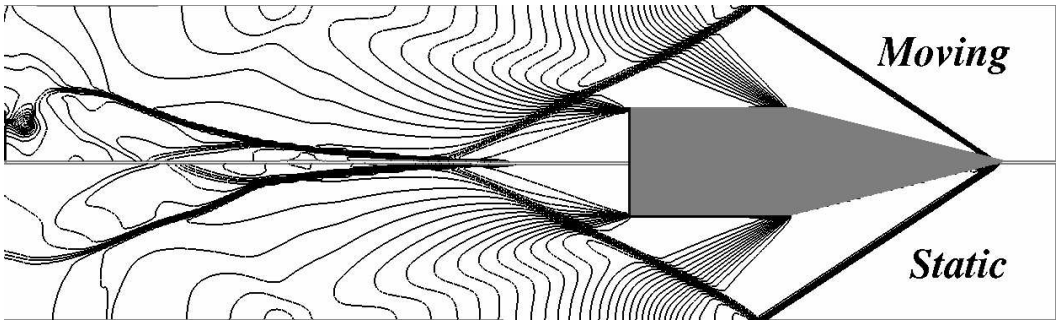


Fig. 6. Supercritical flow past a ship's hull: Depth contours for the moving (port side) and static (starboard side) cases.

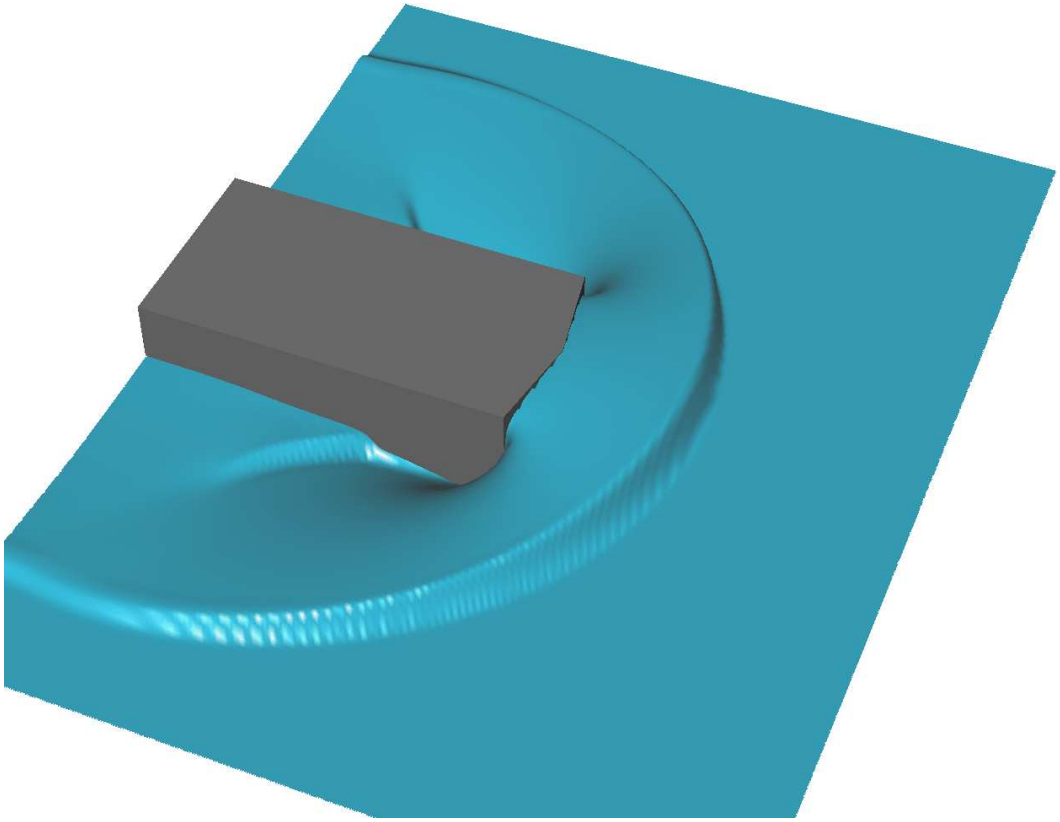


Fig. 7. Hypothetical landslide case: Simulated water surface for the fine grid case.

Table 3
Hypothetical landslide problem: Grid convergence index

point	h_2	h_1	GCI_{21}
$(100.0, 125.0)$	10.0	10.0	0.0
$(100.0, 82.0)$	22.4	22.4	0.2
$(100.0, 95.0)$	21.7	21.8	0.4
$(57.0, 12.0)$	9.92	9.96	0.4
$(150.0, 12.0)$	10.2	10.2	0.0
$(130.0, 55.0)$	6.21	6.00	3.3
$(45.0, 55.0)$	13.0	12.9	0.6
Δx	1.00m	2.00m	

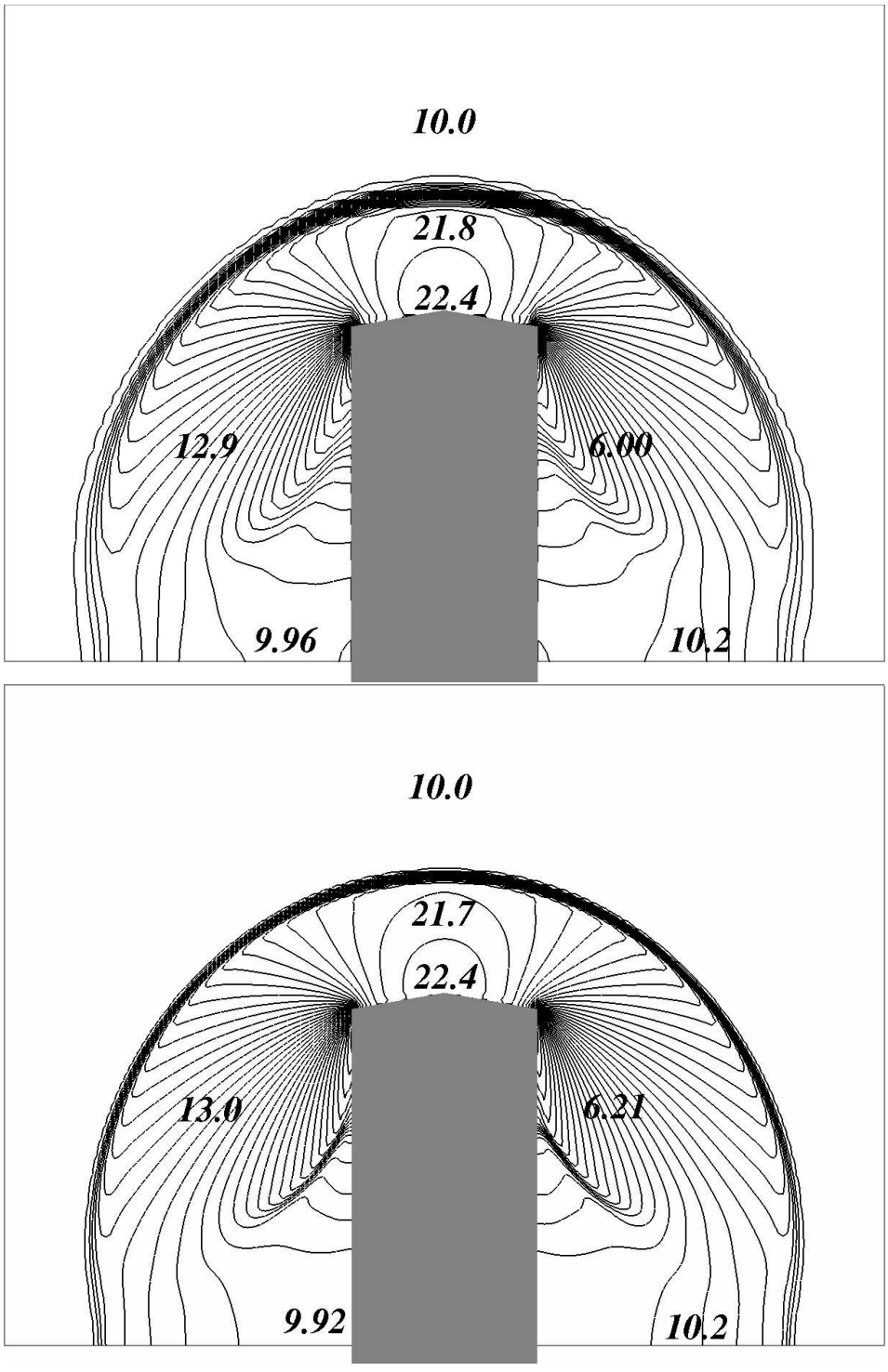


Fig. 8. Hypothetical landslide case: Water depth contours for the coarse (top) and fine (bottom) grid cases showing the values used to compute the GCIs.

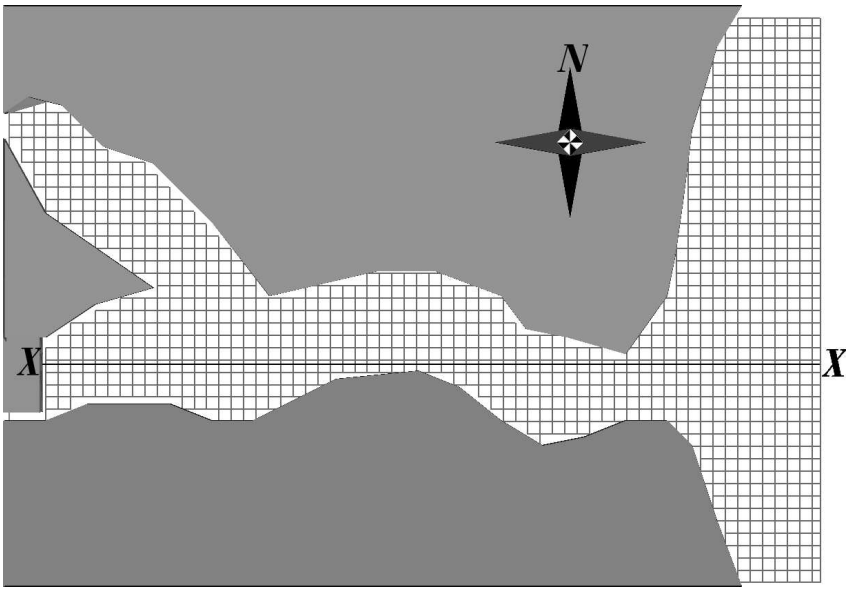


Fig. 9. Landslide in a Fjord: Grid layout at the end of the landslide event (every 4th gridline is shown).

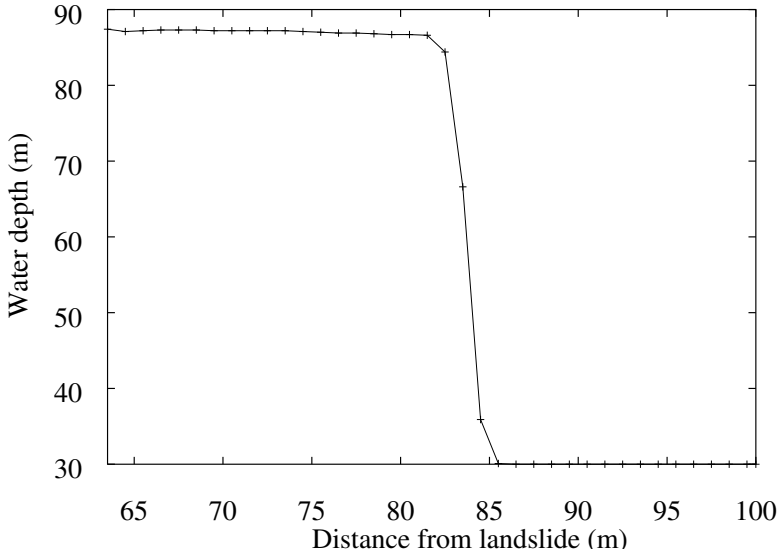


Fig. 10. Landslide in a Fjord: Landslide generated wave (one dimensional test)

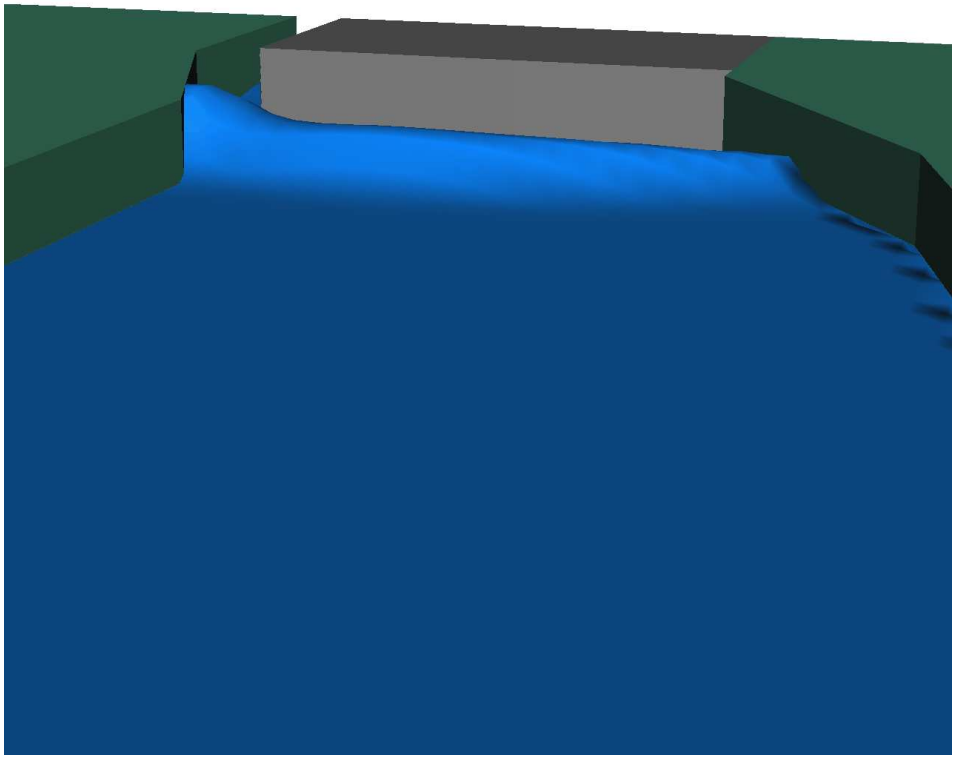


Fig. 11. Landslide in a Fjord: Water surface in the southern branch of the fjord 10 seconds after the start of the landslide event.

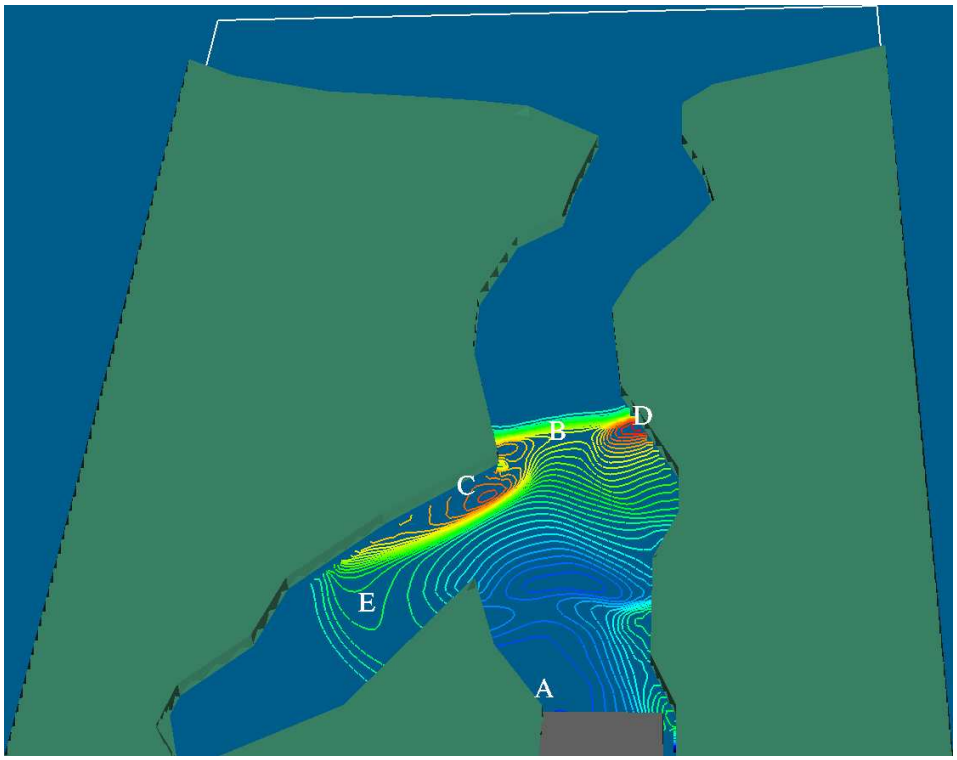


Fig. 12. Landslide in a Fjord: Contours of water depth, looking seaward (east), 60s after the start of the landslide.

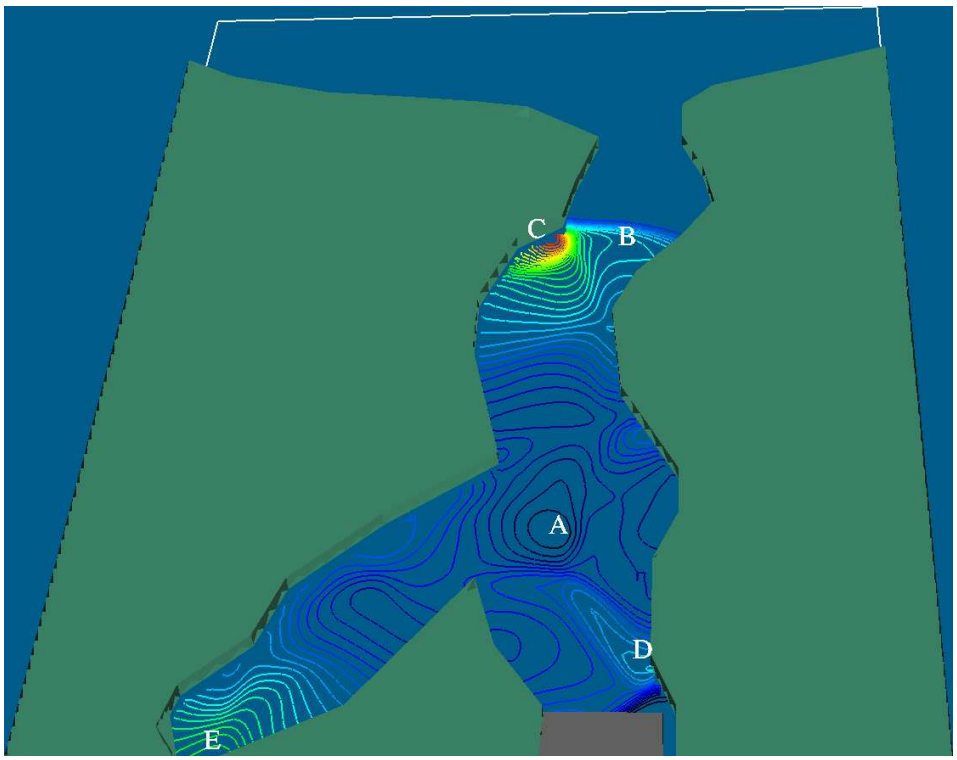


Fig. 13. Landslide in a Fjord: Contours of water depth, looking seaward (east), 120s after the start of the landslide.

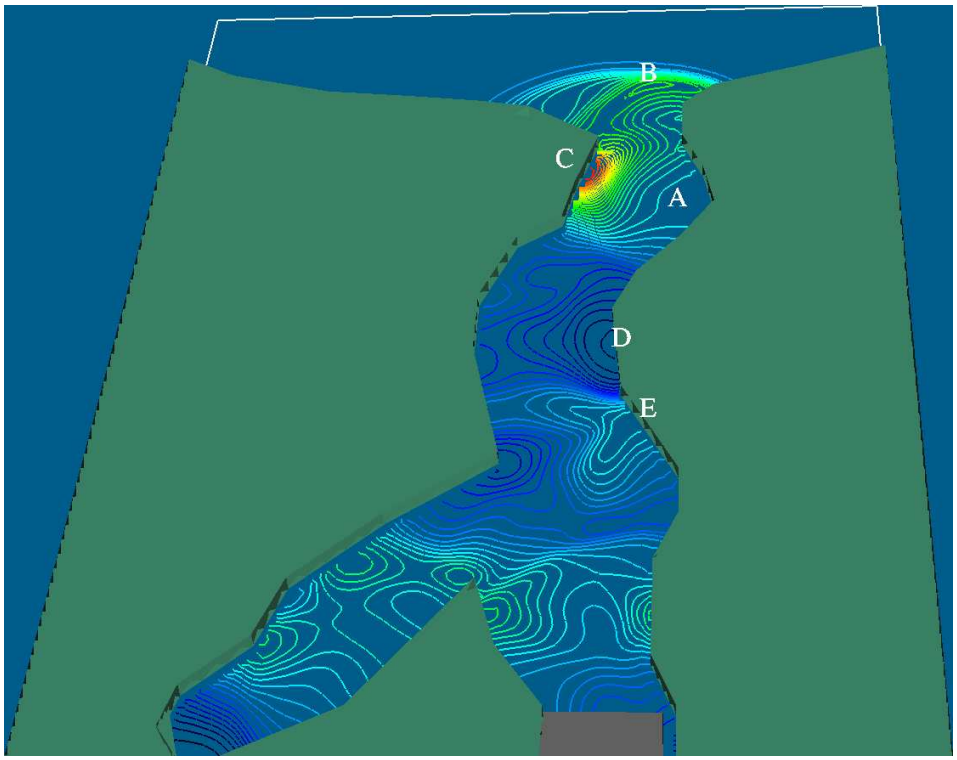


Fig. 14. Landslide in a Fjord: Contours of water depth, looking seaward (east), 180s after the start of the landslide.

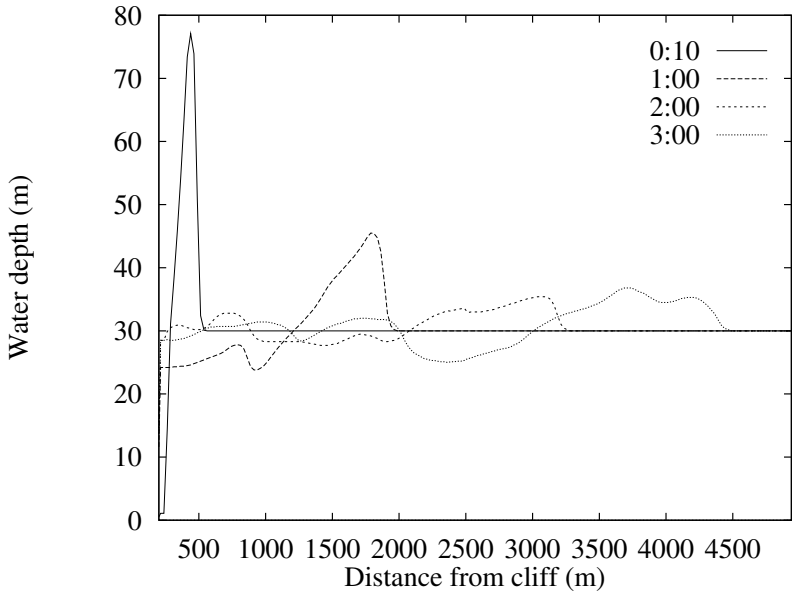


Fig. 15. Landslide in a Fjord: water depth along the main channel (section XX, see Fig. 9)